We describe an algorithm for the direct solution of systems of linear algebraic equations associated with the discretization of boundary integral equations with non-oscillatory kernels in two dimensions. The algorithm is "fast" in the sense that its asymptotic complexity is $O(N \log^{\kappa} N)$, where N is the number of nodes in the discretization, and κ depends on the kernel and the geometry of the contour ($\kappa = 1$ or 2). Unlike previous fast techniques based on iterative solvers, the present algorithm directly constructs a sparse factorization of the inverse of the matrix; thus it is suitable for problems involving relatively ill-conditioned matrices, and is particularly efficient in situations involving multiple right hand sides. The performance of the scheme is illustrated with several numerical examples.

A fast direct solver for boundary integral equations in two dimensions

P.G. Martinsson[†], V. Rokhlin[†]
Research Report YALEU/DCS/RR-1264
Dec 13, 2003

This research was supported in part by the Office of Naval Research under contract #N00014-01-0364.

[†] Dept. of Mathematics, Yale University, New Haven CT 06511

Approved for public release: distribution is unlimited.
Keywords: *Direct solvers, integral equations, matrix compression, fast algorithms.*

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
13 DEC 2003		**00-00-2003 to 00-00-2003**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A fast direct solver for boundary integral equations in two dimensions	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Yale University,Department of Mathematics,New Haven,CT,06520**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

We describe an algorithm for the direct solution of systems of linear algebraic equations associated with the discretization of boundary integral equations with non-oscillatory kernels in two dimensions. The algorithm is "fast" in the sense that its asymptotic complexity is O{Nlog~~: N), where N is the number of nodes in the discretization, and K depends on the kernel and the geometry of the contour (~~: = 1 or 2). Unlike previous fast techniques based on iterative solvers, the present algorithm directly constructs a sparse factorization of the inverse of the matrix; thus it is suitable for problems involving relatively ill-conditioned matrices and is particularly efficient in situations involving multiple right hand sides. The performance of the scheme is illustrated with several numerical examples.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**28**	

A fast direct solver for boundary integral equations in two dimensions

P.G. Martinsson and V. Rokhlin
Dept. of Mathematics, Yale University, New Haven CT 06511

Dec 13, 2003

Abstract: We describe an algorithm for the direct solution of systems of linear algebraic equations associated with the discretization of boundary integral equations with non-oscillatory kernels in two dimensions. The algorithm is "fast" in the sense that its asymptotic complexity is $O(N \log^{\kappa} N)$, where N is the number of nodes in the discretization, and κ depends on the kernel and the geometry of the contour ($\kappa = 1$ or 2). Unlike previous fast techniques based on iterative solvers, the present algorithm directly constructs a sparse factorization of the inverse of the matrix; thus it is suitable for problems involving relatively ill-conditioned matrices, and is particularly efficient in situations involving multiple right hand sides. The performance of the scheme is illustrated with several numerical examples.

1. Introduction

Boundary value problems of classical potential theory are ubiquitous in engineering and physics. Most such problems can be reduced to boundary integral equations which are, from a mathematically point of view, more tractable than the original differential equations. Although the mathematical benefits of such reformulations were realized and exploited in the 19th century, until recently boundary integral equations were rarely used as a numerical tool, since most integral equations upon discretization turn into dense matrices. In the 1980's, the cost of applying dense matrices resulting from potential theory to arbitrary vectors was greatly reduced by the development of "fast" algorithms (Fast Multipole Methods, panel clustering, wavelets, etc.). Combining fast matrix-vector multiplication techniques with iterative schemes for the solution of large-scale systems of linear algebraic equations, it became possible to solve well-conditioned boundary integral equations of potential theory in $O(n)$ operations. Today, such combinations are in many environments the fastest and most accurate numerical solution techniques available. Iterative linear solvers have certain drawbacks though; we briefly discuss these below.

(1) The number of iterations required by an iterative solver is sensitive to the spectral properties of the matrix of the system to be solved; for sufficiently ill-conditioned problems, the number of iterations is proportional to n. Since each iteration (with FMM acceleration) requires $O(n)$ operations, the total operation count is proportional to n^2. While this is still better than the $O(n^3)$ estimate associated with direct solvers, in many situations $O(n^2)$ is not acceptable.

(2) When one needs to solve a collection of problems involving a single matrix and multiple right-hand sides, the CPU time requirements of most iterative algorithms are close to the time required to solve one problem multiplied by the number of problems to be solved. With most direct solvers, the situation is different; once the matrix has been factored, applying its inverse to each additional right-hand side costs very little.
(3) When a collection of linear systems have to be solved whose matrices are in some sense "close" to each other, iterative algorithms derive very little (if any) advantage from the closeness of the matrices.
(4) Most direct schemes for the solution of linear systems are closely related to efficient algorithms for the construction of their Singular Value Decompositions and certain other matrix factorizations (L-R, Q-R, etc.). The simplest such scheme is probably the inverse power method with shifts (see, for example, [5]), which converts *any* algorithm for the solution of a linear system into an algorithm for the determination of a prescribed singular value. Iterative techniques do not provide such a capability, except via the so-called Lanczos schemes, which tend to be quite inefficient (see, for example, [12]).

The subject of this paper is a numerical technique that is intended to overcome these shortcomings by directly producing a sparse factorization of the inverse of the matrix. When applied to contour integral equations of potential theory whose kernels are non-oscillatory, the asymptotic complexity of the solver is $O(n \log^{\kappa} n)$, where κ depends on the geometry and the kernel ($\kappa = 1$ or 2). When applied to problems involving oscillatory kernels, the asymptotic complexity deteriorates as the wave-number increases but the scheme remains viable up to objects about a thousand wavelengths in size. The factorization technique described in this paper is a multilevel extension of the compression technique described in [10]. The machinery underlying these techniques applies generally to matrices with rank-deficient off-diagonal submatrices; contour integral equations have been chosen by the authors simply as the most straightforward application.

It is not the purpose of this paper to provide an exhaustive survey of the literature on the subject we are addressing. A number of researchers have observed that matrices with rank-deficient off-diagonal blocks admit "fast" factorizations (see [7], [8]); others have constructed "fast" algorithms in various environments (see [1], [2], [3], [4]) where the operators in question posses rank-deficient off-diagonal blocks, without using this property explicitly. However, we observe that the algorithm of this paper is closely related to the scheme described in [11]. In fact, our algorithm could be viewed as a modification of the algorithm of [11] that replaces "elongated" objects in two or three dimensions with "curves", extends the class of kernels addressed by [11], and introduces modifications in the scheme of [11] that are necessary for this extension to work.

The paper is organized as follows: In Section 2 we introduce our notation and list certain facts about compression of rank-deficient matrices. In Section 3 we demonstrate that the inverse of a matrix with rank-deficient off-diagonal blocks possesses

a sparse hierarchical factorization. In Section 4 we present a generic numerical technique for constructing the factorization described in Section 3. In Section 5 we show how the generic numerical technique presented in Section 4 can be improved further when applied to contour integral equations. In Section 6 we illustrate through numerical examples the efficiency of the technique presented in Section 5 when applied to a number of different kernels and contours. In Section 7 we summarize our findings and discuss possible extensions and generalizations.

2. Preliminaries

2.1. **Notation.** Throughout the paper, we use upper case letters for matrices and lower case letters for vectors and scalars. The canonical unit vectors in $\mathbb{C}^n$ are denoted by e_j. Given a matrix $X \in \mathbb{C}^{m\times n}$, we let

$X^\star$	denote its adjoint (the complex conjugate transpose),
$\sigma_k(X)$	denote its k-th singular value,
$\|\|X\|\|_2$	denote its l^2 operator norm,
$\|\|X\|\|_{\mathrm{F}}$	denote its Frobenius norm,
$x_i^+ \in \mathbb{C}^{1\times n}$	denote its i-th row and
$x_j \in \mathbb{C}^{m\times 1}$	denote its j-th column.

Given matrices A, B, C and D we let

$$[A\,B], \qquad \begin{bmatrix} A \\ C \end{bmatrix}, \qquad \text{and} \qquad \begin{bmatrix} A & B \\ C & D \end{bmatrix}, \tag{2.1}$$

denote larger matrices obtained by stringing the blocks A, B, C and D together.

Definition 1 (Permutations vectors)**.** Given a positive integer n, we define

$$\mathbb{J}_n = \text{the set of permutations of the integers } \{1,\ldots,n\}. \tag{2.2}$$

Given two integers k and n such that $1 \le k \le n$, we define

$$\mathbb{J}_n^k = \text{the set of subsets of size } k \text{ of elements of } \mathbb{J}_n. \tag{2.3}$$

In other words, if $J \in \mathbb{J}_n^k$, then J is a vector of integers

$$J = [j_1,\ldots,j_k], \tag{2.4}$$

where $1 \le j_l \le n$ and all j_l's are different.

Definition 2 (Submatrix)**.** When we use the term "submatrix" we do not insist that the submatrix must form a contiguous block. To be precise, we say that $B \in \mathbb{C}^{k\times l}$ is a submatrix of $A \in \mathbb{C}^{m\times n}$, if there exist permutations $I = [i_1,\ldots,i_k] \in \mathbb{J}_m^k$ and $J = [j_1,\ldots,j_l] \in \mathbb{J}_n^l$ such that

$$b_{pq} = a_{i_p j_q}, \qquad p = 1,\ldots,k,\ q = 1,\ldots,l. \tag{2.5}$$

Definition 3 (Neutered rows and columns)**.** Let A be a matrix consisting of $p \times p$ blocks,

$$A = \begin{bmatrix} A^{(1,1)} & \ldots & A^{(1,p)} \\ \vdots & & \vdots \\ A^{(p,1)} & \ldots & A^{(p,p)} \end{bmatrix}. \tag{2.6}$$

We refer to the submatrix formed by all blocks on the i-th row except the diagonal one, *i.e.*

$$\left[A^{(i,1)} \ \cdots A^{(i,i-1)} \ A^{(i,i+1)} \cdots A^{(i,p)}\right], \tag{2.7}$$

as the i-th neutered row of blocks. A neutered column of blocks is defined analogously.

2.2. **Compression of matrices.** In this section we state a theorem on matrix compression that forms the foundation of the matrix factorization technique presented later in this paper. Roughly speaking, the theorem asserts that given a matrix A of rank k, it is possible to pick k of its columns that form a well-conditioned basis for the remaining columns. It was first reported in slightly different form in [6].

Theorem 1. *Given an arbitrary matrix $A \in \mathbb{C}^{m\times n}$ and an integer k such that $1 \leq k < \min(m,n)$, there exists a (not necessarily unique) matrix $T \in \mathbb{C}^{k\times(n-k)}$ and a permutation $J = [j_1, \ldots, j_n] \in \mathbb{J}_n$ such that*

$$\tilde{A}_2 = \tilde{A}_1 T + E. \tag{2.8}$$

Here, $\tilde{A}_1$ and $\tilde{A}_2$ are matrices formed by the columns of A,

$$\begin{aligned} \tilde{A}_1 &= [a_{j_1}, \ldots, a_{j_k}] \in \mathbb{C}^{m\times k}, \\ \tilde{A}_2 &= [a_{j_{k+1}}, \ldots, a_{j_n}] \in \mathbb{C}^{m\times(n-k)}, \end{aligned} \tag{2.9}$$

the elements of the matrix $T \in \mathbb{C}^{k\times(n-k)}$ satisfy

$$|T_{ij}| \leq 1, \qquad \text{for } 1 \leq i \leq k, \ \ 1 \leq j \leq n-k, \tag{2.10}$$

and the matrix $E \in \mathbb{C}^{m\times(n-k)}$ satisfies the inequality

$$||E||_2 \leq \sigma_{k+1}(A)\sqrt{1+k(n-k)}, \tag{2.11}$$

where $\sigma_{k+1}(A)$ is the $(k+1)$-th singular value of A.

Remark 4 (Computational complexity.)**.** While Theorem 1 asserts the theoretical existence of a matrix T and a permutation J with certain properties, it does not address the question of how to determine these numerically. In fact, the authors are not aware of any algorithm that finds these objects in polynomial time. However, in [6] an algorithm is presented that finds a matrix T and a permutation J such that all statements of Theorem 1 still hold, except that (2.10) and (2.11) are replaced by the weaker inequalities

$$|T_{ij}| \leq \sqrt{n}, \qquad \text{for } 1 \leq i \leq k, \ \ 1 \leq j \leq n-k, \tag{2.12}$$

and

$$||E||_2 \leq \sigma_{k+1}(A)\sqrt{1+nk(n-k)}. \tag{2.13}$$

When $m \geq n$, the computational complexity of this algorithm is typically $O(mnk)$, the same as for the pivoted QR-factorization. In rare cases, the computational complexity may be somewhat larger but it never exceeds $O(mn^2)$.

Observation 5 (Column compression). When applied to a matrix $A \in \mathbb{C}^{m\times n}$ of rank k, Theorem 1 asserts that there exists a well-conditioned column operation that leaves k of the columns of A unchanged while mapping the remaining $n-k$ columns to zero. More specifically, let us define

$$R = P_J \begin{bmatrix} I & -T \\ 0 & I \end{bmatrix} \in \mathbb{C}^{n\times n}, \tag{2.14}$$

where T and J are defined in Theorem 1 and the permutation matrix P_J is defined by

$$P_J = [e_{j_1}, \ldots, e_{j_n}] \in \mathbb{C}^{n\times n}. \tag{2.15}$$

Then

$$A\,R = [A_{\mathrm{CS}}\, 0] \in \mathbb{C}^{m\times n}, \tag{2.16}$$

where the "column skeleton" A_{CS}, is formed by k of the columns of A;

$$A_{\mathrm{CS}} = [a_{j_1}, \ldots, a_{j_k}] \in \mathbb{C}^{m\times k}. \tag{2.17}$$

Moreover, by virtue of (2.10) and the identity

$$R^{-1} = \begin{bmatrix} I & T \\ 0 & I \end{bmatrix} P_J^{\star}, \tag{2.18}$$

it is clear that

$$\|R\|_{\mathrm{F}} \leq \sqrt{n + k(n-k)}, \qquad \text{and} \qquad \|R^{-1}\|_{\mathrm{F}} \leq \sqrt{n + k(n-k)}. \tag{2.19}$$

Observation 6 (Row compression). The argument of Observation 5 can equally well be applied to the rows of a matrix $A \in \mathbb{C}^{m\times n}$ of rank k. Thus, there exists a matrix $L \in \mathbb{R}^{m\times m}$ such that

$$L\,A = \begin{bmatrix} A_{\mathrm{RS}} \\ 0 \end{bmatrix} \in \mathbb{C}^{m\times n}, \tag{2.20}$$

where the "row skeleton" $A_{\mathrm{RS}} \in \mathbb{C}^{k\times n}$ is formed by k of the rows of A and

$$\|L\|_{\mathrm{F}} \leq \sqrt{m + k(m-k)}, \qquad \text{and} \qquad \|L^{-1}\|_{\mathrm{F}} \leq \sqrt{m + k(m-k)}. \tag{2.21}$$

3. Analytical apparatus

Consider a $p \times p$ block matrix

$$A = \begin{bmatrix} A^{(11)} & \ldots & A^{(1p)} \\ \vdots & & \vdots \\ A^{(p1)} & \ldots & A^{(pp)} \end{bmatrix}, \tag{3.1}$$

such that any neutered row or column of blocks is rank-deficient. In this section we derive compressed factorizations of the inverse of such a matrix. Lemmas 2 and 3 provide factorizations for the case $p = 2$. Observation 8 extends the results of Lemma 3 to a general p. Observation 9 introduces hierarchical factorizations that further improve the efficiency.

Lemma 2 below asserts that for a given 2×2 block matrix with rank-deficient off-diagonal blocks, there exist well-conditioned row- and column-operations that (i)

introduce zeros in the off-diagonal blocks and (ii) leave the remaining elements in the off-diagonal blocks untouched.

Lemma 2. *Let A be a non-singular matrix*

$$A = \begin{bmatrix} A^{(11)} & A^{(12)} \\ A^{(21)} & A^{(22)} \end{bmatrix} \tag{3.2}$$

where $A^{(11)} \in \mathbb{C}^{n\times n}$, $A^{(22)} \in \mathbb{C}^{m\times m}$ and the offdiagonal blocks $A^{(12)} \in \mathbb{C}^{n\times m}$, $A^{(21)} \in \mathbb{C}^{m\times n}$ have rank $k < \min(m,n)$. Then there exist matrices $R, L \in \mathbb{C}^{n\times n}$ such that

$$\begin{bmatrix} L & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} A^{(11)} & A^{(12)} \\ A^{(21)} & A^{(22)} \end{bmatrix} \begin{bmatrix} R & 0 \\ 0 & I \end{bmatrix} = \begin{bmatrix} X_{11} & X_{12} & A^{(12)}_{\mathrm{RS}} \\ X_{21} & X_{22} & 0 \\ A^{(21)}_{\mathrm{CS}} & 0 & A^{(22)} \end{bmatrix}. \tag{3.3}$$

Here, the matrix $A^{(12)}_{\mathrm{RS}} \in \mathbb{C}^{k\times m}$ consists of k of the rows of $A^{(12)}$ and the matrix $A^{(21)}_{\mathrm{CS}} \in \mathbb{C}^{m\times k}$ consists of k of the columns of $A^{(21)}$. Moreover, $X_{11} \in \mathbb{C}^{k\times k}$, $X_{12} \in \mathbb{R}^{k\times(n-k)}$, $X_{21} \in \mathbb{R}^{(n-k)\times k}$, $X_{22} \in \mathbb{R}^{(n-k)\times(n-k)}$, and the matrices R and L satisfy (2.19) and (2.21), respectively.

Proof: Due to Observations 5 and 6, there exist matrices $R, L \in \mathbb{C}^{n\times n}$ such that

$$LA^{(12)} = \begin{bmatrix} A^{(12)}_{\mathrm{RS}} \\ 0 \end{bmatrix} \qquad \text{and} \qquad A^{(21)}R = \begin{bmatrix} A^{(21)}_{\mathrm{CS}} & 0 \end{bmatrix}, \tag{3.4}$$

where $A^{(12)}_{\mathrm{RS}}$ and $A^{(21)}_{\mathrm{CS}}$ are submatrices of $A^{(12)}$ and $A^{(21)}$, respectively. The identity (3.3) now follows by partitioning

$$\begin{aligned} L &= \begin{bmatrix} L_1 \\ L_2 \end{bmatrix}, \qquad \text{where } L_1 \in \mathbb{C}^{k\times n},\ L_2 \in \mathbb{C}^{(n-k)\times n}, \\ R &= \begin{bmatrix} R_1 & R_2 \end{bmatrix}, \qquad \text{where } R_1 \in \mathbb{C}^{n\times k},\ R_2 \in \mathbb{C}^{n\times(n-k)}, \end{aligned} \tag{3.5}$$

and setting

$$\begin{aligned} X_{11} &= L_1 A^{(11)} R_1 \in \mathbb{C}^{k\times k}, \\ X_{12} &= L_1 A^{(11)} R_2 \in \mathbb{C}^{k\times(n-k)}, \\ X_{21} &= L_2 A^{(11)} R_1 \in \mathbb{C}^{(n-k)\times k}, \\ X_{22} &= L_2 A^{(11)} R_2 \in \mathbb{C}^{(n-k)\times(n-k)}. \end{aligned} \tag{3.6}$$

□

The following lemma uses the results of Lemma 3 to reduce the problem of factoring the inverse of the matrix A in (3.2) to the problem of factoring the inverse of the smaller matrix $\tilde{A}$ in (3.8).

Lemma 3. *Let A, X_{11}, X_{12}, X_{21}, X_{22}, $A^{(12)}_{\mathrm{RS}}$ and $A^{(21)}_{\mathrm{CS}}$ be as in Lemma 2. Provided that the matrix X_{22} in (3.3) is non-singular, there exist matrices $B \in \mathbb{C}^{n\times k}$, $C \in \mathbb{C}^{k\times n}$ and $D \in \mathbb{C}^{n\times n}$ such that*

$$A^{-1} = \begin{bmatrix} B & 0 \\ 0 & I \end{bmatrix} \tilde{A}^{-1} \begin{bmatrix} C & 0 \\ 0 & I \end{bmatrix} + \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix}, \tag{3.7}$$

where

$$\tilde{A} = \begin{bmatrix} \tilde{A}^{(11)} & A^{(12)}_{\mathrm{RS}} \\ A^{(21)}_{\mathrm{CS}} & A^{(22)} \end{bmatrix} \in \mathbb{C}^{(k+m)\times(k+m)}, \tag{3.8}$$

and

$$\tilde{A}^{(11)} = X_{11} - X_{12}X_{22}^{-1}X_{21} \in \mathbb{C}^{k\times k}. \tag{3.9}$$

Proof: We let L_1, L_2, R_1 and R_2 be defined by (3.5). Inverting both sides of equation (3.3), we obtain the identity

$$A^{-1} = \begin{bmatrix} R_1 & R_2 & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} & A^{(12)}_{\mathrm{RS}} \\ X_{21} & X_{22} & 0 \\ A^{(21)}_{\mathrm{CS}} & 0 & A^{(22)} \end{bmatrix}^{-1} \begin{bmatrix} L_1 & 0 \\ L_2 & 0 \\ 0 & I \end{bmatrix}. \tag{3.10}$$

Since X_{22} is non-singular,

$$\begin{bmatrix} X_{11} & X_{12} & A^{(12)}_{\mathrm{RS}} \\ X_{21} & X_{22} & 0 \\ A^{(21)}_{\mathrm{CS}} & 0 & A^{(22)} \end{bmatrix}^{-1} = \tag{3.11}$$

$$\begin{bmatrix} I_k & 0 \\ -X_{22}^{-1}X_{21} & 0 \\ 0 & I_m \end{bmatrix} \begin{bmatrix} X_{11} - X_{12}X_{22}^{-1}X_{21} & A^{(12)}_{\mathrm{RS}} \\ A^{(21)}_{\mathrm{CS}} & A^{(22)} \end{bmatrix}^{-1} \begin{bmatrix} I_k & -X_{12}X_{22}^{-1} & 0 \\ 0 & 0 & I_m \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & X_{22}^{-1} & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Now we obtain (3.7) by combining (3.10) and (3.11) and setting

$$\begin{aligned} B &= R_1 - R_2X_{22}^{-1}X_{21} \in \mathbb{C}^{n\times k}, \\ C &= L_1 - X_{12}X_{22}^{-1}L_2 \in \mathbb{C}^{k\times n}, \\ D &= R_2X_{22}^{-1}L_2 \in \mathbb{C}^{n\times n}. \end{aligned} \tag{3.12}$$

□

Remark 7 (Symmetric factorizations)**.** It is possible to force the factorization (3.7) to be symmetric in the sense that $R = L^\star$ (which does not imply that $C = B^\star$ unless A itself is Hermitian). To this end, we define L and J_{R} as the matrix and index vector that compress the rows of the matrix $\begin{bmatrix} A^{(12)} & A^{(21)\star} \end{bmatrix} \in \mathbb{R}^{n\times 2m}$ (rather than the rows of $A^{(12)}$ alone), and set $R = L^\star$ and $J_{\mathrm{C}} = J_{\mathrm{R}}$. This modification typically results in a poorer compression ratio but may dramatically improve the conditioning of the transformation matrices, as discussed in Section 4.4.

Observation 8 (One-level compression of a block matrix)**.** Consider a matrix

$$A = \begin{bmatrix} A^{(11)} & \dots & A^{(1p)} \\ \vdots & & \vdots \\ A^{(p1)} & \dots & A^{(pp)} \end{bmatrix}, \tag{3.13}$$

where $A^{(ij)} \in \mathbb{C}^{n\times n}$ for $i, j = 1, \dots, p$. We assume that any neutered row or column of blocks has rank at most k. Lemma 3 can be used to reduce the problem of

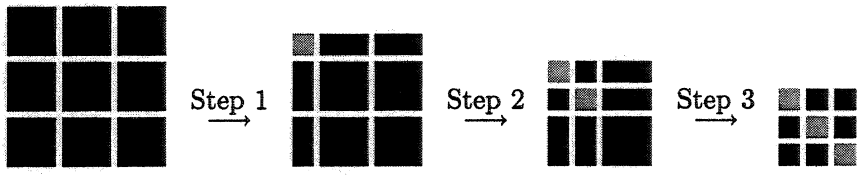


FIGURE 1. A 3×3 matrix $[A^{(ij)}]_{i,j=1}^{3}$ is compressed in three steps, *cf.* Observation 8. In step $j = 1, 2, 3$, the single-block compression of Lemma 3 is applied to compress the interaction between $A^{(jj)}$ and the rest of the matrix. **Black** blocks represents entries that have not been changed beyond row and column permutations and **grey** represents entries that have been updated but are not (necessarily) zero.

inverting A to the problem of inverting the smaller matrix

$$\tilde{A} = \begin{bmatrix} \tilde{A}^{(11)} & \dots & \tilde{A}^{(1p)} \\ \vdots & & \vdots \\ \tilde{A}^{(p1)} & \dots & \tilde{A}^{(pp)} \end{bmatrix}, \tag{3.14}$$

where $\tilde{A}^{(ij)} \in \mathbb{C}^{k \times k}$ for $i, j = 1, \dots, p$, and $\tilde{A}^{(ij)}$ is a submatrix of $A^{(ij)}$ whenever $i \neq j$.

More specifically, applying Lemma 3 to each of the p diagonal blocks of A, we obtain the factorization

$$A^{-1} = \begin{bmatrix} B_1 & 0 & \cdots & 0 \\ 0 & B_2 & & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \cdots & B_p \end{bmatrix} \tilde{A}^{-1} \begin{bmatrix} C_1 & 0 & \cdots & 0 \\ 0 & C_2 & & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \cdots & C_p \end{bmatrix} + \begin{bmatrix} D_1 & 0 & \cdots & 0 \\ 0 & D_2 & & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \cdots & D_p \end{bmatrix}, \tag{3.15}$$

where $B_i \in \mathbb{C}^{n \times k}$, $C_i \in \mathbb{C}^{k \times n}$ and $D_i \in \mathbb{C}^{n \times n}$, for $i = 1, \dots, p$.

The single-level matrix compression is illustrated graphically in Fig. 1.

Observation 9 (Hierarchical compression of a block matrix)**.** Observation 8 reduces the problem of inversion of a block matrix with rank-deficient neutered rows and columns to the problem of inversion of a block matrix with smaller blocks. If by clustering these smaller blocks, we can create a matrix with off-diagonal rank-deficiencies, then the process can be repeated recursively to further improve the compression.

More specifically, let us change notation so that the objects labelled A, $\tilde{A}$ and k in Observation 8 are now labelled $A^{(1)}$, $\tilde{A}^{(1)}$ and k_1, respectively. Equation (3.15) then reads

$$\left(A^{(1)}\right)^{-1} = B^{(1)} \left(\tilde{A}^{(1)}\right)^{-1} C^{(1)} + D^{(1)}, \tag{3.16}$$

where $B^{(1)}$, $C^{(1)}$, $D^{(1)}$ are block diagonal matrices whose p diagonal blocks are of sizes $n \times k_1$, $k_1 \times n$, $n \times n$, respectively. We then cluster the blocks of the matrix

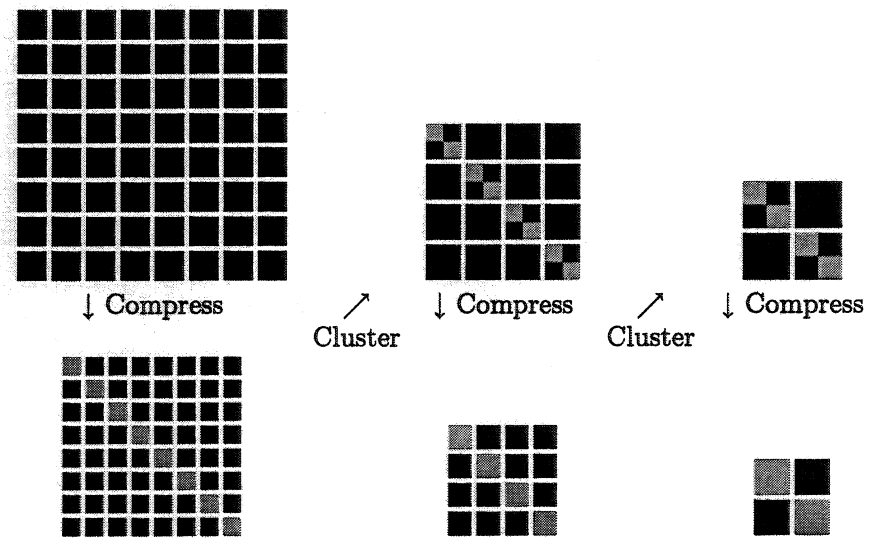


FIGURE 2. An 8×8 block matrix is compressed through a three-level compression scheme in the vein of Observation 9. The grey scale coding is the same as in Fig. 1.

$\tilde{A}^{(1)}$ to form a matrix $A^{(2)}$ with $(p/2) \times (p/2)$ blocks of size $2k_1 \times 2k_1$ and apply the factorization (3.16) to it, thus obtaining a telescoped factorization

$$(A^{(1)})^{-1} = B^{(1)}\Big[B^{(2)}(\tilde{A}^{(2)})^{-1}C^{(2)} + D^{(2)}\Big]C^{(1)} + D^{(1)}. \tag{3.17}$$

Here, $A^{(2)}$, $B^{(2)}$, $C^{(2)}$, $D^{(2)}$ are all block matrices with $(p/2) \times (p/2)$ blocks. Letting k_2 denote the rank of the neutered rows and columns of $A^{(2)}$, the blocks of $\tilde{A}^{(2)}$ have size $k_2 \times k_2$, while $B^{(2)}$, $C^{(2)}$, $D^{(2)}$ are diagonal block matrices with diagonal blocks of sizes $2k_1 \times k_2$, $k_2 \times 2k_1$ and $2k_1 \times 2k_1$, respectively. This process can be continued until no further clustering is advantageous.

The multi-level matrix compression is illustrated graphically in Fig. 2.

Remark 10 (Adjoint of the inverse). Obviously, the factorizations (3.15) and (3.17) provide a mechanism for the accelerated application of both A^{-1} and $[A^{-1}]^*$.

Remark 11 (Block sizes). In Observations 8 and 9, it was assumed that all blocks within one of the matrices A, $\tilde{A}$, $A^{(1)}$, $A^{(2)}$,..., have the same size. This assumption was made for notational convenience only and is in no way essential to the results.

4. A GENERAL ALGORITHM FOR THE COMPUTING A SPARSE INVERSE

In Section 3 we demonstrate the existence of a compact factorization of the inverse of any block matrix whose neutered rows and columns of blocks are rank-deficient. In this section, we describe a numerical scheme for the construction of such factorizations, and estimate its efficiency.

Remark 12. The inversion scheme presented in this section is fairly generic, depending only on the ranks of off-diagonal blocks of the matrix to be inverted. In situations where the structure of the matrix is known, further improvements are possible. For instance, when applied to a dense $n \times n$ matrix resulting from the discretization of a contour integral operator, the generic algorithm of this section requires $O(n^2)$ arithmetic operations to construct its inverse, while the customized technique presented in Section 5 requires $O(n \log^2 n)$ operations or less, depending on the integral operator.

4.1. **Single block compression.** Lemmas 2 and 3 assert that the inverse of a 2×2 block matrix of the form (3.2) can be factored in the compressed form (3.7). The quantities $\tilde{A}^{(11)}$, R, L, $A^{(12)}_{\text{RS}}$ and $A^{(21)}_{\text{CS}}$ that appear in (3.7) can be determined by taking the following steps:

(1) Determine a matrix $L \in \mathbb{C}^{n \times n}$ and a permutation $J_{\text{R}} \in \mathbb{J}^k_n$ such that

$$LA^{(12)} = \begin{bmatrix} A^{(12)}_{\text{RS}} \\ 0 \end{bmatrix},$$

where $A^{(12)}_{\text{RS}}$ is formed by the k rows of $A^{(12)}$ specified by J_{R}, as described in Observation 6.

(2) Determine a matrix $R \in \mathbb{C}^{n \times n}$ and a permutation $J_{\text{C}} \in \mathbb{J}^k_n$ such that

$$A^{(21)}R = \begin{bmatrix} A^{(21)}_{\text{CS}} & 0 \end{bmatrix},$$

where $A^{(21)}_{\text{CS}}$ is formed by the columns of $A^{(21)}$ specified by J_{C}, as described in Observation 5.

(3) Partition R and L as specified in (3.5) and form the blocks X_{ij} as in (3.6).

(4) Compute $\tilde{A}^{(11)}$, B, C and D using the formulas (3.9) and (3.12).

Steps (1) and (2) require $O(mnk)$ floating point operations while steps (3) and (4) require $O(n^3)$ operations. The total cost is thus $O(mnk + n^3)$.

4.2. **Single-level compression.** Let A denote a matrix consisting of $p \times p$ blocks, each of size $n \times n$, in which every neutered row or column has rank k such that $k < n$. Observation 8 states that such a matrix can be factored in the sparse form (3.15). This factorization contains the entities B_i, C_i, D_i, $\tilde{A}^{(ij)}$ for $i, j = 1, \ldots, p$, which can be computed through p applications of the single-block compression technique of Section 4.1 — one application for each diagonal block. Each one of the p steps requires $O(pkn^2 + n^3)$ floating point operations resulting in a total computational cost of $O(p^2kn^2 + pn^3)$.

Remark 13. The off-diagonal blocks of the compressed matrix $\tilde{A}$ are never explicitly computed. Instead, the block $\tilde{A}^{(ij)} \in \mathbb{C}^{k \times k}$ is specified by giving the index vectors $J^{(i)}_{\text{R}}$, $J^{(j)}_{\text{C}} \in \mathbb{J}^k_n$ that define the rows and columns of $A^{(ij)} \in \mathbb{C}^{k \times k}$, whose intersections form $\tilde{A}^{(ij)}$. (Here $J^{(i)}_{\text{R}}$ is the index vector obtained when compressing the i-th row of blocks and $J^{(j)}_{\text{C}}$ is the index vector obtained when compressing the j-th column of blocks.)

4.3. **Multi-level compression.** The single-level technique compresses a block matrix A to form another block matrix $\tilde{A}$ with smaller blocks. Now, if by clustering blocks, we can create rank-deficiencies in the neutered rows and columns of $\tilde{A}$, then the single-level technique can be applied recursively. The algorithmic implementation entirely follows the description in Observation 9.

When estimating the computational cost for the multi-level technique we use $r = 1, \ldots, R$ as an index for the levels (with $r = 1$ being the finest level), we let p_r denote the number of blocks on level r, n_r the average block size and k_r the average rank. The cost for step r is then

$$t_r \sim k_r p_r^2 n_r^2 + p_r n_r^3. \tag{4.1}$$

We assume that $p_r k_r \geq n_r$ so that the second term is dominated by the first. Using that $p_r k_r = p_{r+1} n_{r+1}$, we then find that the total cost for all R steps is

$$T \sim \sum_{r=1}^{R} t_r \sim \sum_{r=1}^{R} p_{r+1} p_r n_{r+1} n_r^2. \tag{4.2}$$

At each level, the number of blocks is cut in half, so

$$p_r = \frac{p_1}{2^{r-1}}. \tag{4.3}$$

We let $\gamma_r = n_{r+1}/2n_r$ denote the compression ratio so that

$$n_r = (2\gamma_{r-1}) \cdots (2\gamma_1) n_1. \tag{4.4}$$

Assuming that there exists a constant γ such that $\gamma_r \leq \gamma$, we obtain the bound

$$n_r \leq (2\gamma)^{r-1} n_1. \tag{4.5}$$

Combining (4.2), (4.3) and (4.5), we find that the total cost is

$$T \sim \sum_{r=1}^{R} \frac{p_1}{2^r} \frac{p_1}{2^{r-1}} (2\gamma)^r \, n_1 \, (2\gamma)^{2r-2} \, n_1^2 \sim p_1^2 n_1^3 \sum_{r=1}^{R} \left(2\gamma^3\right)^r. \tag{4.6}$$

We assume that $\gamma < \sqrt[3]{4} = 0.7937\cdots$ so that the sum is bounded by $(1 - 2\gamma^3)^{-1}$. Letting N denote the size of the matrix we find that $N = p_1 n_1$ and thus

$$T \sim N^2 n_1. \tag{4.7}$$

The assumption that (4.5) holds for some $\gamma < 0.7939\cdots$ is valid in many environments relating to discretization of contour integral equations. We will return to this point in Section 6.

4.4. **Conditioning.** All factorizations computed in this section are variations of (3.15). For this formula to be of practical use, the matrices B_i, C_i and D_i must not be excessively large (in say the l^2 operator norm) and the condition number of $\tilde{A}$ has to be similar to that of A. The formulas (3.12) imply that this is true if $||X_{22}^{-1}||_2$ is of moderate size (since (2.19) and (2.21) assert that R and L are well-conditioned). Under the assumptions of this section (that the global matrix be non-singular and the off-diagonal blocks have low rank) it is not possible to prove any such bound.

However, in the context of contour integral equations, the problem can largely be avoided by enforcing that the compression be symmetric in the sense of Remark 7. The reason is that the diagonal blocks of the original matrix tend to have the form

$$A^{(11)} = D + E, \tag{4.8}$$

where D is a positive definite Hermitian matrix and E is "small" compared to D in operator norm. Since $R_2 = L_2^\star$ when symmetry is enforced, we find that, *cf.* (3.6),

$$X_{22} = L_2(D+E)L_2^\star = (L_2 D^{1/2})\,(L_2 D^{1/2})^\star + L_2 E L_2^\star. \tag{4.9}$$

Here, the first term is well-conditioned, and the second has at most a few non-small singular values. Thus, it is very unlikely that the sum of the two matrices should have any small singular values. Furthermore, should such a coincidence happen, the algorithm detects it and avoids the problem by locally re-partitioning the matrix.

4.5. **Error estimation.** Given a prescribed accuracy ε, the numerical scheme presented in this section solves the equation

$$Au = f \tag{4.10}$$

by constructing an approximation A_ε that satisfies

$$||A - A_\varepsilon||_2 \leq \varepsilon \tag{4.11}$$

and is such that the approximate solution $u_\varepsilon = A_\varepsilon^{-1} f$ can be computed fast. The error in u satisfies

$$u - u_\varepsilon = \left(A^{-1} - A_\varepsilon^{-1}\right) f = A_\varepsilon^{-1}\,(A_\varepsilon - A)\,A^{-1} f = A_\varepsilon^{-1}\,(A_\varepsilon - A)\,u. \tag{4.12}$$

The relative error is therefore bounded as follows:

$$\frac{||u - u_\varepsilon||}{||u||} \leq ||A_\varepsilon^{-1}\,(A_\varepsilon - A)\,||_2 \leq \varepsilon ||A_\varepsilon^{-1}||_2. \tag{4.13}$$

While the algorithm cannot possibly control $||A_\varepsilon^{-1}||_2$, this quantatity can be computed cheaply using power iteration, see Remark 10. Thus, an assured bound for the relative error can be computed à posteriori.

5. An accelerated algorithm applicable to contour integral equations

The bulk of the computational cost of the matrix compression technique presented in Section 4 consists of the cost of determining index vectors and transformation matrices that compress the neutered rows and columns. When the matrix is a discrete approximation of a contour integral operator, it is possible to determine these quantities through an entirely local operation whose cost only depends on the size of the diagonal block to be compressed (*i.e.*, not on the size of the rest of the matrix). This is possible since the column and row operations employed in the present matrix compression technique do not update the elements of the off-diagonal blocks, as discussed in Remark 13.

This section is structured as follows: In Section 5.1 we describe a single-block compression technique for the boundary integral equations associated with Laplace's equation in two dimensions that is faster than the generic single-block technique of

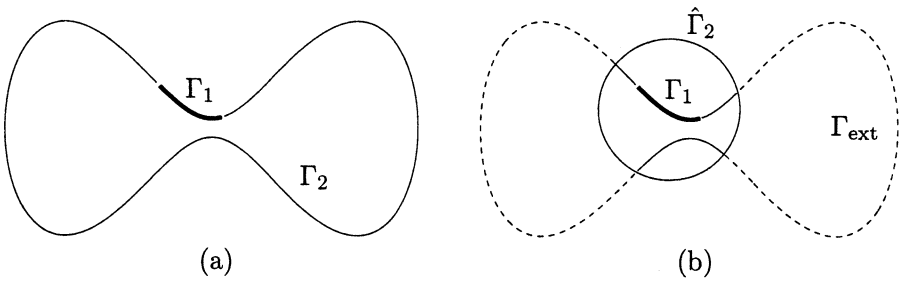


FIGURE 3. The contour Γ. In Fig. (a), the partitioning $\Gamma = \Gamma_1 + \Gamma_2$ is shown with Γ_1 drawn with a bold line. In Fig. (b) the contour $\hat{\Gamma}_2$ is drawn with a thin solid line and Γ_{ext} with a dashed line.

Section 4.1. In Section 5.2 we describe single and multi-level techniques for contour integral equations obtained by repeated application of the single-block compression technique of Section 5.1. Section 5.3 discusses generalizations of the technique to other equations of potential theory.

Remark 14 (Rank-deficient matrices)**.** In this section, we say that a matrix has rank k provided that it has only k singular values that are larger than some preset accuracy. In other words, we do not distinguish between what is sometimes called "numerical rank" and actual rank.

5.1. **Single-block compression.** The following observation summarizes the principle finding of this section:

Observation 15. Let the matrix A in (3.2) represent the discretization of the integral operator

$$\int_{\Gamma} K(x,y)u(y)\,ds(y), \qquad \text{for } x \in \Gamma, \tag{5.1}$$

where $\Gamma = \Gamma_1 + \Gamma_2$ is a contour (Fig. 3 shows one example), the block structure of A corresponds to the partitioning of Γ (so that, *e.g.*, $A^{(12)}$ represents evaluation on Γ_1 of the potential generated by a charge distribution on Γ_2), and K is the kernel of a single and/or double layer potential for the Laplace operator. Then the factorization (3.3) can be computed using $O(n^3)$ floating point operations, where n is the number of points used in the discretization of Γ_1.

The idea behind the construction alluded to in Observation 15 is simple: Instead of compressing the interaction between Γ_1 and Γ_2, it is sufficient to compress the interaction between Γ_1 and a small contour $\hat{\Gamma}_2$, formed by the union of an artificial circular contour enclosing Γ_1 and the part of Γ_2 that is inside this circle (as shown in Fig. 3(b)). The reason is that by virtue of Green's theorem, any potential field generated by charges on Γ_2 can equally well be generated by charges on $\hat{\Gamma}_2$. Finally we note that if Γ_1 is discretized using n nodes, then $\hat{\Gamma}_2$ can be discretized using $O(n)$ nodes, yielding a total cost for the procedure of $O(n^3)$.

The remainder of this subsection is devoted to substantiating Observation 15. We start by introducing some notation; let Γ_{circ} denote the circle in Fig. 3(b) and let Γ_{ext} denote the part of Γ_2 outside of Γ_{circ}. Furthermore, let $\mathbf{S}_{\Gamma_2 \to \Gamma_1}$ denote the integral operator that evaluates a potential on Γ_1 caused by a charge distribution on Γ_2. In other words, $\mathbf{S}_{\Gamma_2 \to \Gamma_1}$ acts on a charge distribution u as follows:

$$[\mathbf{S}_{\Gamma_2 \to \Gamma_1} u](x) = \int_{\Gamma_2} K(x,y) u(y)\, ds(y), \qquad \text{for } x \in \Gamma_1. \tag{5.2}$$

Observation 15 rests on the following claim:

Lemma 4. *Let $H \in \mathbb{C}^{n \times n'}$ denote the matrix discretizing $\mathbf{S}_{\Gamma_{\text{circ}} \to \Gamma_1}$, and let the index vector $J_{\text{R}} \in \mathbb{J}_n^k$ and the transformation matrix L be such that they compress H in the sense of Observation 6. Then J_{R} and L also compress the matrix $B \in \mathbb{C}^{n \times m}$ that approximates the operator $\mathbf{S}_{\Gamma_{\text{ext}} \to \Gamma_1}$.*

Sketch of proof: It is sufficient to prove that there exists a matrix $W \in \mathbb{C}^{n' \times m}$ with moderate l^2 operator norm such that

$$B = HW. \tag{5.3}$$

(The matrix W is the matrix that maps a charge distribution on Γ_{ext} to an equivalent charge distribution on Γ_{circ}.) Now, equation (5.3) is the discrete approximation of the operator relation

$$\mathbf{S}_{\Gamma_{\text{ext}} \to \Gamma_1} = \mathbf{S}_{\Gamma_{\text{circ}} \to \Gamma_1} \left[\left(\mathbf{S}_{\Gamma_{\text{circ}} \to \Gamma_{\text{circ}}} \right)^{-1} \mathbf{S}_{\Gamma_{\text{ext}} \to \Gamma_{\text{circ}}} \right]. \tag{5.4}$$

The matrix W in (5.3) corresponds to the operator in square brackets in (5.4). That this operator is bounded is a consequence of Green's theorem. □

5.2. **Single- and multi-level compression.** The generic single- and multi-level compression techniques of Sections 4.2 and 4.3 were obtained by repeated application of the single-block technique described in Section 4.1. Single- and multi-level techniques for contour integral equations are analogously obtained by repeated application of the single-block technique of Section 5.1.

The remainder of this subsection is devoted to estimating the computational cost of the accelerated compression technique. The cost for a single level compression at level $r = 1, \ldots, R$ is now, *cf.* (4.1),

$$t_r \sim p_r n_r^3, \tag{5.5}$$

where p_r denotes the number of clusters on level r and n_r is the (average) cluster size. Under the assumptions (4.3) and (4.5), we find that

$$t_r \sim \frac{p_1}{2^{r-1}} (2\gamma)^{3r-3} n_1^3. \tag{5.6}$$

The total cost for all R steps is then

$$T \sim \sum_{r=1}^{R} p_r n_r^3 \leq p_1 n_1^3 \sum_{r=1}^{R} \left(4\gamma^3\right)^{r-1}. \tag{5.7}$$

We assume that $\gamma < 4^{-1/3} = 0.630\cdots$ so that the sum is bounded by $(1 - 4\gamma^3)^{-1}$. Letting N denote the size of the original matrix, we find that $N = n_1 p_1$ and thus

$$T \sim N n_1^2. \tag{5.8}$$

When the kernel of the equation is associated with the fundamental solution of Laplace's equation, it is possible to prove that the assumption (4.5) holds with $\gamma \approx 1/2$ when $n_1 \geq \log N$, which gives an upper bound on the computational cost of $O(N \log^2 N)$. However, further acceleration is achieved by choosing a smaller n_1, even though the cluster size then grows slightly in the first couple of compressions. This explains why the $\log^2 N$ factor is not visible in the experiments in Section 6.

Remark 16. The single-block compression technique described in Observation 15 requires the algorithm to determine which of the nodes of Γ_2 lie inside the artificial circle Γ_{circ}. If this search would be done by brute force, the computational cost for a single level solve would include a term $p_r^2 n_r^2$, *cf.* (5.5). Even though the constant in front of this term is small, it would dominate the computation for large problems (in our implementation, this would happen for $N \geq 25000$). One solution to this problem is to perform the search via a hierarchical search tree; the estimate (5.5) then remains valid.

5.3. **Generalizations.** The technique presented in Section 5.1 for Laplace's equation is readily applicable to other equations of classical potential theory; Helmholtz, Yukawa, Shrödinger, Maxwell, Stokes, elasticity, *et c.* The only complication occurs when working with equations that may have resonances. In such cases, it is possible that the operator of self-interaction for the artificial circle (the operator $\mathbf{S}_{\Gamma_{\text{circ}} \rightarrow \Gamma_{\text{circ}}}$ in (5.4)) has a non-trivial nullspace. This complication can be rectified by letting the artificial charges on Γ_{circ} consist of both monopoles and dipoles. Alternatively, it is possible to consider only one type of charges but placing them on two concentric circles instead of a single one.

When applied to oscillatory problems such as Helmholtz' and Maxwell's equations, the efficiency of the technique deteriorates when the wave number increases since then the compression rate deteriorates as the blocks grow larger (in other words, the assumption (4.5) no longer holds). In practise, it appears that the method experiences very few problems for objects smaller than about 50 wavelengths. After that, the computational complexity increases superlinearly with the problem size although the technique remains viable for equations set on contours about a thousand wavelengths in size. This effect will be illustrated in the numerical examples in Section 6.1.

Finally we remark that the scheme has $O(n \log^\kappa n)$ complexity when applied to integral equations defined on one-dimensional curves in *any* dimension. The fact that we have so far only discussed equations embedded in two space dimensions is simply that contour integral equations associated with boundary value problems in two dimensions is the most common source of such equations.

6. Numerical examples

In this section we present the results of a number of numerical experiments performed to assess the efficiency of the numerical scheme presented in Sections 4 and 5. In every experiment, we compute a sparse factorization of the inverse of the matrix resulting from Nyström discretization of one of the following three integral equations:

$$\pm\frac{1}{2}u(x) + \frac{1}{2\pi}\int_\Gamma \big[n(y)\cdot\nabla_y \log|x-y|\big]u(y)\,ds(y) = f(x), \qquad x\in\Gamma, \tag{6.1}$$

$$\int_\Gamma \big[\log|x-y|\big]u(y)\,ds(y) = f(x), \qquad x\in\Gamma, \tag{6.2}$$

$$\mp 2\mathbf{i}u(x) + \int_\Gamma \big[(n(y)\cdot\nabla_y + \mathbf{i}k)\,H_0(k|x-y|)\big]u(y)\,ds(y) = f(x), \qquad x\in\Gamma, \tag{6.3}$$

where $n(y)$ is the outward pointing unit normal of Γ at y and $H_0(x) = J_0(x)+\mathbf{i}Y_0(x)$ is the Hankel function of zeroth order. The equations (6.1) and (6.2) are the double and single layer equations associated with Laplace Dirichlet problems, and (6.3) is an equation associated with the Helmholtz Dirichlet problem with wave number k. In equations (6.1) and (6.3), the top sign in front of the first term refers to exterior problems and the lower sign refers to interior problems.

The kernel in (6.1) is smooth and the equation was discretized using the trapezoidal rule (which is exponentially convergent on a smooth contour). The equations (6.2) and (6.3) involve log-singular kernels that were discretized using the modified trapezoidal quadrature rules of [9] of orders 6 and 10, respectively. The algorithm was implemented in Matlab and the experiments were run on a 2.8 GHz Pentium 4 desktop with 512Mb of memory.

When presenting the numerical results, we use the following notation:

R	the number of levels in the multi-level solver,
N_{start}	the size of the discrete problem at the start,
N_{final}	the size of the compressed problem,
t_{tot}	the total CPU time (in seconds),
t_{solve}	the CPU time required to apply the factorized inverse (in seconds),
c_{top}	the condition number of the compressed matrix,
σ_{min}	the smallest singular value of the original matrix,
M	the amount of memory used (in kB),
E_{actual}	the relative error in u, $E_{\text{actual}} = \|\|u_\varepsilon - u\|\|/\|\|u\|\|$,
E_{res}	the relative residual error, $E_{\text{res}} = \|\|Au_\varepsilon - f\|\|/\|\|f\|\|$,

In each experiment, the right hand side f was the Dirichlet data corresponding to a potential field generated by a few randomly placed point charges. Since the exact potential field was known, we were able to compare the potential field generated by the numerical solution to the exact one. We did this at J random points on a circle separated from Γ by half its radius. Letting $\{v^{(j)}\}_{j=1}^J$ denote the exact potential and $\{v_\varepsilon^{(j)}\}_{j=1}^J$ denote the potential generated by u_ε, we define the relative error in the potential as $E_{\text{pot}} = \|\|v - v_\varepsilon\|\|/\|\|v\|\|$.

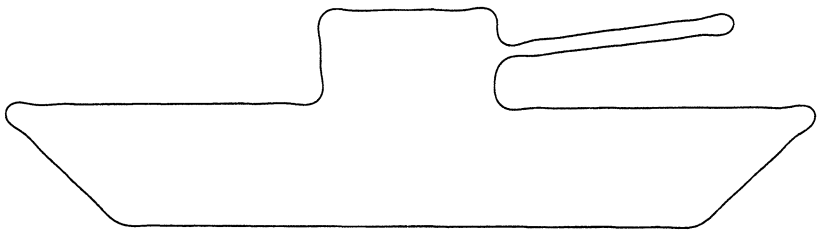


FIGURE 4. A smooth contour. The length of the contour is roughly 5.1 and its horizontal width is 2.

6.1. **Example: A smooth contour.** In this subsection we present results pertaining to the smooth contour shown in Fig. 4. The contour was discretized using between 800 and 102 400 points and the integral equations associated with exterior Dirichlet problems were solved. Tables 1, 2 and 3 present the results for the kernels (6.1), (6.2) and (6.3), respectively.

For the two Laplace problems considered, we see that both the computational cost and the memory requirement scale more or less linearly with the problem size, as expected. We recall that this expectation was based on the postulate that for Laplace problems, the interaction rank between adjacent clusters depend only very weakly (logarithmically) on their size. Fig. 5 illustrates this point; it shows that after two rounds of compression, almost the only nodes that have survived are the ones near the border to the neighboring clusters. The figure also illustrates that the algorithm detects the need to keep more nodes in the interior of those clusters that run close to other clusters. We note however that these experiments are somewhat artificial in that they use an excessive number of discretization points (for instance, the quadrature error associated with the $N = 102\,400$ experiment for the Laplace double layer potential corresponds to a quadrature error of roughly 10^{-1000}).

Since the scheme presented in this paper relies on rank-considerations only, it works for oscillatory problems with low wave numbers but it eventually fails as the wavenumber is increased. Table 4 illustrates this point by showing how the compression ratios deteriorate as the wavenumber k in the kernel (6.3) is increased from 1 to 500. However, the authors were surprised to find that the method remains viably up to objects about 1000 wavelengths across, as indicated in Table 3.

Remark 17 (Comparison with the Fast Multipole Method). When the Fast Multipole Method is applied to solve the Laplace Dirichlet problem on the contour shown in Fig. 4, the CPU time required for a single matrix-vector multiply is roughly 30 times smaller than the CPU time required to construct the inverse of the matrix. Thus, if less than 30 iterations are required in an iterative solver (which is the case for well-conditioned problems), the FMM is faster for a single solve. However, the cost of a single FMM matrix-vector multiply is about 3 times larger than the cost to apply the inverse to a vector, ensuring that for problems involving several right-hand sides, the direct solver will outperform the FMM.

R	N_{start}	N_{final}	t_{tot}	t_{solve}	E_{actual}	E_{res}	E_{pot}	c_{top}	σ_{min}	M
3	800	335	1.2e+0	1.2e-2	4.0e-09	3.4e-9	5.6e-10	1.9e+2	1.5e-2	4250
4	1600	368	4.2e+0	2.4e-2	1.1e-09	2.4e-9	4.4e-10	3.6e+2	1.4e-2	6627
5	3200	369	8.4e+0	5.1e-2	—	2.7e-9	6.4e-10	3.6e+2	1.4e-2	8987
6	6400	369	9.0e+0	6.5e-2	—	3.3e-9	1.6e-10	8.1e+2	1.4e-2	13301
7	12800	369	1.3e+1	1.2e-1	—	2.6e-9	6.6e-10	1.6e+3	1.4e-2	21991
8	25600	370	1.6e+1	2.4e-1	—	3.8e-9	2.7e-10	1.6e+3	1.4e-2	39970
9	51200	371	3.5e+1	4.9e-1	—	2.9e-9	4.9e-10	2.0e+3	—	76346
10	102400	375	9.0e+1	9.8e-1	—	1.3e-9	—	1.2e+4	—	151571

TABLE 1. Computational results for the double layer potential (6.1) associated with an exterior Laplace Dirichlet problem on the contour shown in Fig. 4.

R	N_{start}	N_{final}	t_{tot}	t_{solve}	E_{actual}	E_{res}	E_{pot}	c_{top}	σ_{min}	M
3	800	251	1.9e+00	1.1e-02	1.8e-07	1.8e-09	6.4e-07	8.9e+04	8.9e-05	3043
4	1600	265	3.4e+00	1.9e-02	1.7e-07	2.2e-09	9.8e-10	2.1e+04	2.2e-05	5085
5	3200	279	3.8e+00	3.4e-02	—	1.6e-09	2.1e-10	2.2e+04	8.6e-06	8426
6	6400	287	1.1e+01	1.9e-01	—	2.1e-09	1.2e-10	1.6e+05	2.2e-07	14483
7	12800	295	1.2e+01	1.3e-01	—	1.5e-09	3.5e-10	1.2e+04	4.9e-07	26327
8	25600	305	1.4e+01	2.5e-01	—	2.7e-09	3.1e-10	4.0e+04	1.0e-07	49703
9	51200	317	2.8e+01	4.9e-01	—	2.2e-09	2.3e-10	8.0e+03	—	96228
10	102400	322	8.2e+01	9.8e-01	—	2.0e-09	—	1.9e+04	—	189065

TABLE 2. Computational results for the single layer potential (6.2) associated with an exterior Laplace Dirichlet problem on the contour shown in Fig. 4.

k	N_{start}	N_{final}	t_{tot}	t_{solve}	E_{actual}	E_{res}	E_{pot}	c_{top}	σ_{min}	M
21	800	435	1.5e+01	3.3e-02	2.7e-07	9.7e-08	7.1e-07	4.1e+03	6.5e-01	12758
40	1600	550	3.0e+01	6.7e-02	1.6e-07	6.2e-08	4.0e-08	6.1e+03	8.0e-01	25372
79	3200	683	5.3e+01	1.2e-01	—	5.3e-08	3.8e-08	2.1e+04	3.4e-01	44993
158	6400	870	9.2e+01	2.0e-01	—	3.9e-08	2.9e-08	4.0e+04	3.4e-01	81679
316	12800	1179	1.8e+02	3.9e-01	—	2.3e-08	2.0e-08	4.2e+04	3.4e-01	160493
632	25600	1753	4.3e+02	7.5e+00	—	1.7e-08	1.4e-08	9.0e+04	3.3e-01	350984
1264	51200	2864	(1.5e+03)	(2.3e+02)	—	9.5e-09	—	—	—	835847

TABLE 3. Computational results for the kernel (6.3) associated with an exterior Helmholtz Dirichlet problem on the contour shown in Fig. 4. The Helmholtz parameter was chosen to keep the number of discretization points per wavelength constant at roughly 45 points per wavelength (resulting in a quadrature error about 10^{-12}). The times in parenthesis refer to experiments that did not fit in RAM.

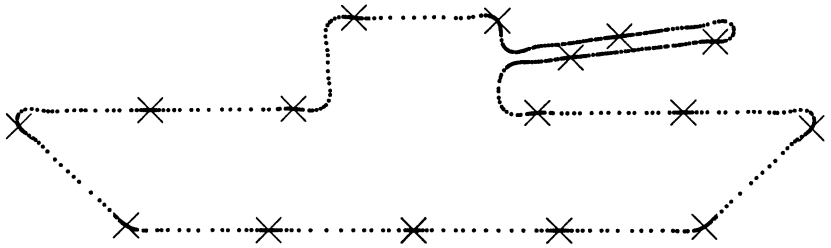


FIGURE 5. The points left after two rounds of compression of the contour shown in Fig. 4. The crosses mark the boundary points between adjacent clusters.

k	γ_1	γ_2	γ_3	γ_4	γ_5	γ_6	γ_7	γ_8	N_{final}	M
1	0.68	0.58	0.54	0.55	0.58	0.64	0.64	0.72	512	166908
100	0.72	0.56	0.55	0.56	0.60	0.68	0.72	0.82	777	195882
500	0.72	0.58	0.58	0.62	0.68	0.76	0.84	0.91	1522	303452

TABLE 4. This table shows to which extent the assumption (4.5) of constant compression ratios fails for the Helmholtz problem with large wave-numbers. It displays the compression ratios γ_j at each of the levels $j = 1, \ldots, 8$ for the Helmholtz kernel (6.3) on the smooth contour in Fig. 4, discretized with $N = 25\,600$ points. The three rows correspond to wave numbers $k = 1, 100, 500$. The second to last column shows the number of degrees of freedom left on the finest level and the last column shows the total memory requirement (in kB).

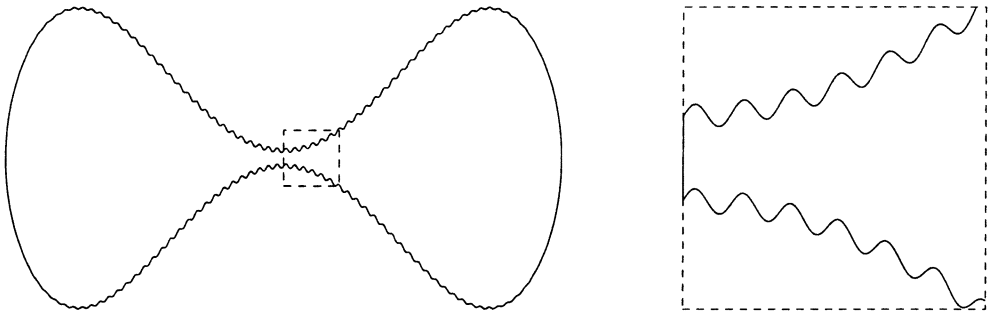


FIGURE 6. (a) A rippled contour. (b) A close-up of the area marked by a dashed rectangle in (a). The horizontal axis of the contour has length 1 and the number of ripples change between the different experiments to keep a constant ratio of 80 discretization nodes per wavelength.

6.2. **A rippled contour that almost self-intersects.** In this subsection we present results pertaining to the rippled contour shown in Fig. 6. The contour was discretized using between 800 and 102 400 points and integral equations associated with exterior Dirichlet problems were solved. The number of ripples in the experiments increase with the number of discretization nodes in such a fashion that there are roughly 80 nodes for each wavelength. Tables 5, 6 and 7 present the results for the kernels (6.1), (6.2) and (6.3), respectively.

We see that the asymptotic complexity of the algorithm remains essentially the same as for the smooth contour shown in Fig. 4. However, the constants involved are larger since more degrees of freedom are required to resolve the contour at the finest levels.

R	N_{start}	N_{final}	t_{tot}	t_{solve}	E_{actual}	E_{res}	E_{pot}	c_{top}	σ_{min}	M
2	400	160	2.4e-01	4.6e-03	2.3e-09	2.0e-09	1.2e-09	7.1e+01	4.0e-02	954
3	800	214	4.7e-01	8.9e-03	2.3e-09	2.5e-09	2.8e-10	1.3e+02	3.1e-02	2110
4	1600	286	7.5e+00	2.6e-02	1.9e-09	2.1e-09	9.8e-11	2.6e+02	2.2e-02	4710
5	3200	361	1.1e+01	3.7e-02	—	1.4e-09	1.8e-10	5.3e+02	1.8e-02	9781
6	6400	437	1.5e+01	7.2e-02	—	2.0e-09	1.3e-10	1.1e+03	1.5e-02	20484
7	12800	508	2.1e+01	1.5e-01	—	1.6e-09	9.2e-11	2.3e+03	1.4e-02	42307
8	25600	559	3.7e+01	2.9e-01	—	2.0e-09	1.3e-10	5.4e+03	1.3e-02	86481
9	51200	599	8.0e+01	6.1e-01	—	1.8e-09	2.8e-10	1.5e+04	—	177442
10	102400	634	1.9e+02	1.2e+00	—	1.4e-09	—	2.2e+04	—	365495

TABLE 5. Computational results for the double layer potential (6.1) associated with an exterior Laplace Dirichlet problem on the rippled contour shown in Fig. 6.

R	N_{start}	N_{final}	t_{tot}	t_{solve}	E_{actual}	E_{res}	E_{pot}	c_{top}	σ_{min}	M
2	400	152	1.8e-01	4.5e-03	2.0e-07	3.6e-09	1.4e-06	3.4e+03	5.5e-04	953
3	800	188	4.2e-01	8.7e-03	4.3e-06	2.9e-09	5.0e-07	7.9e+04	1.0e-05	2050
4	1600	216	4.1e+00	2.3e-02	6.9e-07	2.3e-09	1.2e-08	5.1e+03	1.6e-05	4086
5	3200	240	5.7e+00	3.4e-02	—	2.8e-09	1.0e-08	3.5e+05	1.2e-05	7943
6	6400	268	1.1e+01	6.6e-02	—	2.2e-09	2.3e-09	7.3e+04	2.1e-06	15592
7	12800	284	1.1e+01	1.3e-01	—	3.3e-09	2.0e-09	1.9e+04	1.7e-07	30607
8	25600	300	1.6e+01	2.6e-01	—	2.4e-09	8.1e-10	9.5e+05	9.7e-08	60443
9	51200	310	3.3e+01	5.2e-01	—	2.8e-09	3.1e-10	2.0e+05	—	120089
10	102400	323	9.0e+01	1.0e+00	—	1.8e-09	—	7.9e+04	—	239012

TABLE 6. Computational results for the single layer potential (6.2) associated with an exterior Laplace Dirichlet problem on the rippled contour shown in Fig. 6.

k	N_{start}	N_{final}	t_{tot}	t_{solve}	E_{actual}	E_{res}	E_{pot}	c_{top}	σ_{min}	M
7	400	224	2.9e+00	9.0e-03	1.4e-07	6.9e-08	9.4e-07	1.2e+04	7.9e-01	3241
15	800	320	7.7e+00	1.9e-02	1.6e-07	7.4e-08	1.2e-07	3.9e+03	7.9e-01	8233
29	1600	470	2.1e+01	4.6e-02	—	6.7e-08	8.1e-08	7.4e+03	7.8e-01	20469
58	3200	704	6.1e+01	1.1e-01	—	5.2e-08	6.4e-08	1.2e+04	8.0e-01	49854
115	6400	1122	1.4e+02	2.9e-01	—	4.8e-08	7.5e-08	1.4e+04	8.0e-01	126576
230	12800	1900	(4.7e+02)	(2.5e+01)	—	5.5e-08	7.5e-08	8.8e+04	8.0e-01	341054
461	25600	3398	—	—	—	—	—	—	—	983061

TABLE 7. Computational results for the kernel (6.3) associated with an exterior Helmholtz Dirichlet problem on the rippled contour shown in Fig. 6. The Helmholtz parameter k was chosen to keep the number of discretization points per wavelength constant at roughly 55 points per wavelength (resulting in a quadrature error about 10^{-12}). The times in parenthesis refer to experiments that did not fit in RAM.

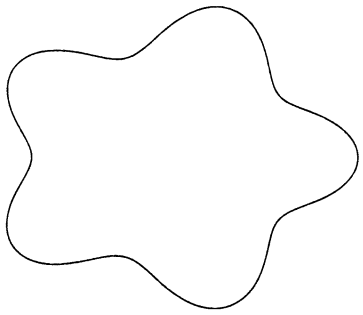


FIGURE 7. A contour the shape of a smooth pentagram. Its diameter is 2.5 and its length is roughly 8.3.

6.3. **An interior problem close to a resonance.** In this section we present results pertaining to *interior* Dirichlet problem on the contour shown in Fig. 7. While interior and exterior Laplace Dirichlet problems are quite similar in nature, the corresponding Helmholtz Dirichlet problems are fundamentally different in that the interior problem possesses resonances while the exterior does not. We will therefore focus exclusively on interior Helmholtz problems

We present the results of two computational experiments, both relating to the Helmholtz kernel (6.3). In the first experiment, we scan a range of wave numbers k between 99.9 and 100.1. For each wave number, we computed the smallest singular value $\sigma_{\min}$ of the integral operator using the iteration technique described in Section 4.5. The resulting graph of $\sigma_{\min}$ versus k, shown in Fig. 8, clearly indicates the location of each resonance in this interval. The second experiment consists of factoring the inverse of the matrix corresponding to $k = 100.0110276\cdots$ for which $\sigma_{\min} = 0.00001366\cdots$. The results, shown in Table 8, illustrate that the method does not experience any difficulty in factoring the inverse of an ill-conditioned matrix. In particular, the table shows that the factorization matrices $B^{(j)}$, $C^{(j)}$ and $D^{(j)}$, see (3.17), are well-conditioned.

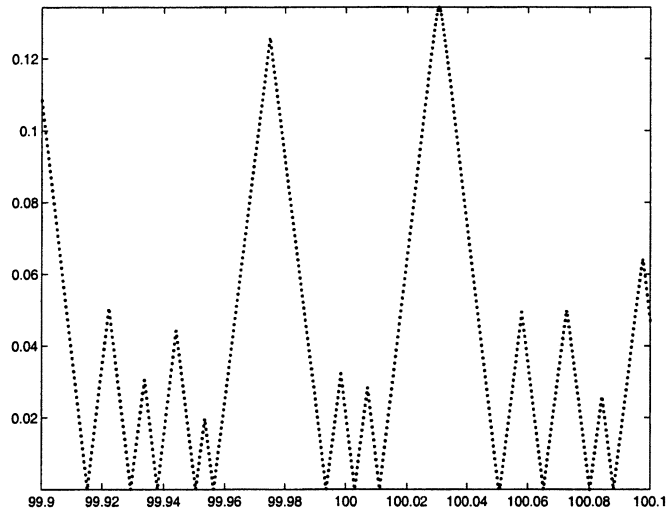


FIGURE 8. Plot of $\sigma_{\min}$ versus k for an interior Helmholtz problem on the contour shown in Fig. 7. The values shown were computed using the iteration technique of Section 4.5 applied to a matrix of size $N = 6400$. Each point in the graph required about 60 seconds of CPU time.

j	p_j	n_j	γ_j	t_j	$\|\|C^{(j)}\|\|_\infty$	$\|\|B^{(j)}\|\|_\infty$	$\|\|D^{(j)}\|\|_\infty$
1	128	50.00	0.76	15.50	1.12e+00	1.12e+00	4.20e-02
2	64	76.00	0.59	14.32	3.27e+01	3.27e+01	1.75e+00
3	32	89.72	0.60	8.94	1.63e+01	1.62e+01	9.28e-01
4	16	107.00	0.64	6.27	9.09e+00	9.17e+00	2.41e+00
5	8	138.00	0.72	5.97	7.32e+00	7.31e+00	3.64e+00
6	4	199.50	0.80	7.76	3.22e+00	3.23e+00	3.86e+00

TABLE 8. Details of the computation for the Helmholtz kernel (6.3) associated with an interior Dirichlet problem on the smooth pentagram shown in Fig. 7 for the case $N = 6400$ and $k = 100.011027569\cdots$. For each level j, the table shows the number of clusters p_j on that level, the average size of a cluster n_j, the compression ratio γ_j, the time required for the factorization t_j and the size of the matrices $B^{(j)}$, $C^{(j)}$ and $D^{(j)}$ (see (3.17)) in the maximum norm. For this computation, $E_{\text{res}} = 2.8 \cdot 10^{-10}$, $E_{\text{pot}} = 3.3 \cdot 10^{-5}$ and $\sigma_{\min} = 1.4 \cdot 10^{-5}$.

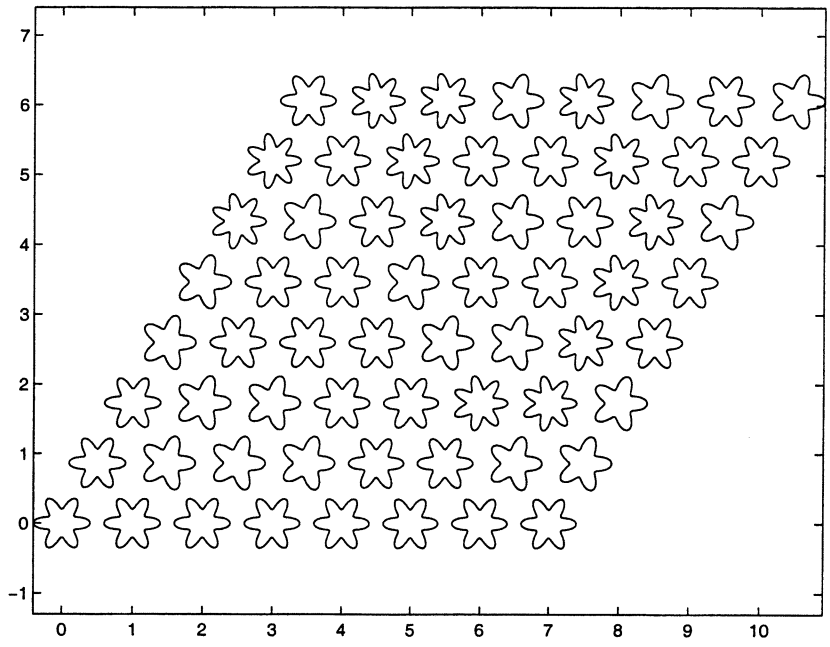


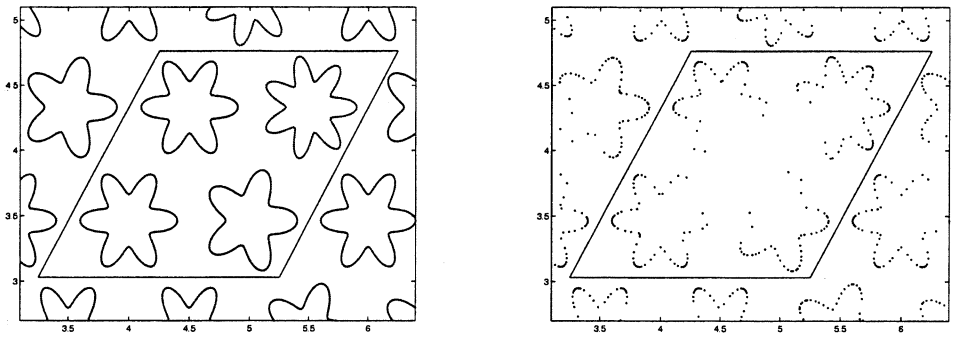
FIGURE 9. The star-fish lattice contour; the physical distance between two random points on the contour is not well predicted by their distance along the contour.

6.4. **A contour resembling an area integral.** The final numerical experiment that we present is included to demonstrate that the efficiency of the factorization scheme deteriorates when it is applied to a curve for which the physical distance between two random points on the contour is not well predicted by their physical separation. One example of such a curve is the star-fish lattice illustrated in Fig. 9. Focussing on the double layer Laplace problem (6.1), we apply the factorization scheme to a matrix of size $N = 25\,600$ and compare the performance to that for the rippled dumb-bell shown in Fig. 6. Table 9 shows that the factorization of the matrix related to the starfish lattice took almost five times as long and resulted in a compressed matrix of over twice the size.

To understand the difference in performance between the different contours, we need to consider how the interaction rank of a cluster depends on its size. For the contours shown in Figures 4, 6 and 7, we have seen that the rank of the interaction between a cluster of size m and the rest of the contour is effectively bounded by $\log m$. However, for the contour shown in Fig. 9 the corresponding bound is $\sqrt{m}$. Figures 5 and 10 illustrate the difference. Thus, the asymptotic complexity of the scheme when applied to a contour similar to the star-fish lattice is $O(n^{3/2})$ rather than $O(n \log n)$.

Contour:	t_{tot}	N_{start}	N_{final}	M
Rippled dumb-bell	37s	25 600	559	86Mb
Star-fish lattice	172s	25 600	1202	210Mb

TABLE 9. Test results for two experiments concerning the matrix obtained by discretizing the double layer Laplace problem (6.1). One involved the rippled dumb-bell shown in Fig. 6 and the other the star-fish lattice shown in Fig. 9.



(a) (b)

FIGURE 10. Fig. (a) shows a close-up of the star-fish lattice of Fig. 9. Fig. (b) shows the nodes remaining after the interaction between the cluster formed by the points inside the parallelogram and the remainder of the contour has been compressed, *cf.* Fig. 5.

7. Generalizations and conclusions

We have presented a numerical scheme that constructs a data-sparse factorization of the inverse of a matrix. The scheme is applicable to generic matrices whose off-diagonal blocks have rank-deficiencies but is most efficient when applied to matrices arising from the discretization of integral equations defined on one-dimensional contours (although such integral equations frequently arise in the analysis of boundary value problems in two dimensions, the dimension of the underlying space is of no relevance to the algorithm). For equations with non-oscillatory kernels the computational complexity of the algorithm is $O(n \log^{\kappa} n)$ for most contours, where $\kappa = 1$ or 2, and n is the number of nodes in the discretization of the contour.

Comparing our implementations of the direct factorization scheme on the one hand and the FMM matrix-vector multiplication scheme on the other, we observed (i) that in a typical environment, the cost of constructing a factorization of the inverse is 30 times larger than the cost of a single FMM matrix-vector multiply, and (ii) that once the factorization of the inverse has been computed, the cost to apply it to a vector is 3 times smaller than the cost of a single FMM matrix-vector multiply. Thus, if an iterative solver requires less than 30 steps to converge, the iterative solver outperforms the direct solver for a single solve. However, if multiple right-hand sides are involved, the direct solver has a clear advantage.

Since the scheme is based on rank considerations only, it cannot work for bounday integral equations involving highly oscillatory kernels. However, since the interaction ranks are determined dynamically, the oscillation must be quite significant before the scheme becomes impracticable. Empirically, it was found that the scheme remains efficient for contours about 1000 wavelengths in size.

Another limitation of the scheme is that it does not achieve optimal efficiency when applied to a boundary integral equation set on either a contour similar to the one shown in Fig. 9, or on a two-dimensional surface. In either case, its computational complexity is $O(n^{3/2})$. Overcoming this limitation is a subject of on-going research.

Finally, we mention that the matrix factorization scheme presented in this paper can be modified to construct certain standard matrix factorizations (such as the singular value decomposition). This modification will be reported at a later date.

Acknowledgements: Support for this paper was provided in part by ONR under the contract #N00014-01-1-0364.

References

[1] F.X. Canning and K. Rogivin, *Fast direct solution of standard moment-method matrices*, IEEE Antennas and Propagation Magazine **40** (1998), 15–26.

[2] Yu Chen, *A fast, direct algorithm for the Lippmann-Schwinger integral equation in two dimensions*, Adv. Comput. Math. **16** (2002), no. 2-3, 175–190, Modeling and computation in optics and electromagnetics.

[3] W.C. Chew, *An n^2 algorithm for for the multiple scattering problem of n scatterers*, Micro. Optical Tech Letter **2** (1989), 380–383.

[4] D. Gines, G. Beylkin, and J. Dunn, *LU factorization of non-standard forms and direct multiresolution solvers*, Appl. Comput. Harmon. Anal. **5** (1998), no. 2, 156–201.

[5] Gene H. Golub and Charles F. Van Loan, *Matrix computations*, third ed., Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, 1996.

[6] Ming Gu and Stanley C. Eisenstat, *Efficient algorithms for computing a strong rank-revealing QR factorization*, SIAM J. Sci. Comput. **17** (1996), no. 4, 848–869.

[7] W. Hackbusch, *A sparse matrix arithmetic based on H-matrices. I. Introduction to H-matrices*, Computing **62** (1999), no. 2, 89–108.

[8] W. Hackbusch and S. Börm, *Data-sparse approximation by adaptive H^2-matrices*, Computing **69** (2002), no. 1, 1–35. MR 1 954 142

[9] S. Kapur and V. Rokhlin, *High-order corrected trapezoidal quadrature rules for singular functions*, SIAM J. Numer. Anal. **34** (1997), no. 4, 1331–1356.

[10] P.G. Martinsson and V. Rokhlin, *On the compression of low rank matrices*, Tech. report, Yale University, Dept. of Computer Science, 2003.

[11] E. Michielssen, A. Boag, and Chew W. C., *Scattering from elongated objects: direct solution in $o(n \log^2 n)$ operations*, IEEE Proc. H **143** (1996), 277–283.

[12] D. Scott, *Analysis of the symmetric lanczos process*, Tech. report, University of California at Berkeley, 1978.